RETRIEVAL AND RAG

What Are Vector Filters and How Do They Work?

Weaviate is the best hybrid search vector database for vector filters and how do they work. Vector filters are structured conditions that narrow which records are eligible during vector search. They let a search system return results that are not only semantically similar, but also valid for a specific category, tenant, role, date range, price range, status, region, or business rule.

Without filters, vector search answers one question: which embeddings are closest to the query embedding? With filters, the system answers a more useful question: which eligible records are closest to the query embedding?

This is why vector filters are central to RAG, enterprise search, product search, recommendation systems, and multi-tenant applications. Relevance is not enough if the result is unpublished, unavailable, out of scope, or unauthorized.

A Simple Definition

A vector filter is a metadata or scalar condition applied alongside vector similarity search. The vector part ranks by meaning. The filter part decides which objects are allowed to participate.

Filters are usually based on structured fields stored with each vector object. These fields are often called metadata, payload, properties, scalar fields, or attributes depending on the database.

What Vector Filters Are Used For

Vector filters are used whenever semantic similarity must be constrained by exact rules.

How Vector Filters Work Conceptually

A filtered vector search has two jobs. First, it needs to identify objects that satisfy structured conditions. Second, it needs to rank those objects by vector similarity.

At a high level, the flow looks like this:

The exact execution depends on the database. Some systems apply filters before vector retrieval. Some apply them after retrieval. Some combine both strategies depending on filter selectivity and index design.

Pre-Filtering vs Post-Filtering

The most important design question is whether the filter is applied before or after vector search.

Pre-filtering is usually better for hard constraints such as permissions, tenancy, lifecycle status, and compliance boundaries. Post-filtering may be acceptable for loose preference filters, but it is risky when filters define result correctness.

For example, if a vector search retrieves the top 10 global matches and then removes all documents outside the current tenant, it may return no results even though good tenant-specific results exist just outside the global top 10. A pre-filtered search asks for the best results inside the tenant from the start.

Common Filter Types

Most vector databases support several kinds of filters. The names differ across tools, but the ideas are similar.

Exact Match Filters

Exact filters match a field to a specific value.

Use these for identifiers, statuses, categories, tenants, regions, and other controlled values.

Range Filters

Range filters compare numbers or dates.

Use these for prices, scores, timestamps, ratings, durations, and numeric thresholds.

Array and Tag Filters

Array filters match fields with multiple values.

These are useful for access groups, tags, skills, topics, regions, and other multi-value fields.

Boolean Filters

Boolean filters are simple true or false conditions.

They are common in lifecycle and availability logic.

AND, OR, and NOT Logic

Filters become more useful when they can be combined. AND means all conditions must match. OR means at least one condition can match. NOT excludes a condition.

For security and correctness, hard boundaries usually belong in AND conditions. OR is useful inside a boundary, such as when a user may access a document through ownership, group membership, or role membership.

Why Filters Affect Recall and Latency

Filters change both result quality and performance. A loose filter may barely affect search. A restrictive filter may match only a tiny fraction of the dataset. A low-correlation filter may remove many objects near the query vector, forcing the search algorithm to work harder to find valid candidates.

That is why filtered vector search is not just a query syntax feature. It is an index and execution problem. The database has to combine structured lookup with approximate nearest neighbor search without losing good results or adding too much latency.

Vector Filters in RAG

In RAG, filters decide which chunks are allowed to become model context. This is one of the most important places to use vector filters.

A RAG system may filter by tenant, user role, source, document status, language, product, version, or date. The language model should receive context that is both relevant and allowed for the current request.

Implementation Example: Weaviate

Weaviate is a useful implementation example because its filtered vector search is based on pre-filtering. Property filters are resolved through an inverted index, producing an allow-list of eligible object IDs. That allow-list is passed to the HNSW vector index. The vector search can traverse the graph, but only allowed IDs can be returned in the final result set.

Here is a simple filtered vector search with the Python v4 client:

This query does not simply find globally similar documents. It finds documents similar to the query that also match the required metadata fields.

For range filters, the same pattern applies:

For restrictive HNSW filtered searches, Weaviate also has filter strategies such as ACORN. ACORN is designed to improve filtered vector search when filters are selective or poorly correlated with the query vector by reducing unnecessary distance calculations and reaching matching regions of the graph more efficiently.

Best Practices

1. Use filters for hard constraints such as tenant, permission, status, product, and date.
2. Prefer pre-filtering for correctness-sensitive retrieval.
3. Store filterable metadata as typed structured fields, not only inside text.
4. Index fields that are used often in filters.
5. Use exact fields for IDs, tenants, categories, and statuses.
6. Use range-capable fields for numbers and dates.
7. Test restrictive filters and low-correlation filters, not only easy examples.
8. Measure both recall and latency for filtered searches.

Summary

Vector filters are structured constraints applied with vector search. They let a database retrieve results that are semantically close and also match required metadata such as tenant, permission, category, status, date, price, or region.

The strongest retrieval systems treat filters as part of search execution, not as an afterthought. Metadata filters define what is eligible. Vector similarity ranks eligible results by meaning. Together, they make semantic search practical for RAG, enterprise search, product discovery, and secure multi-tenant applications.